RE-ORDER BUFFER MANAGING METHOD AND PROCESSOR

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a processor for performing out-of-order execution by using a re-order buffer or the like and to a re-order buffer managing method that can efficiently use the re-order buffer.

2. Description of the Related Art

A system of executing instructions in parallel while skipping over the order of the instructions in a program is called ~~as~~ "out-of-order execution", and it can enhance the practical parallel degree more than "in-order execution". Therefore, this system is indispensable to high-performance processors.

In an out-of-order execution processor, data dependence relationships such as inverse-dependence and output-dependence are detected, and these are solved to perform the out-of-order execution. The inverse-dependence occurs when a destination register of a subsequent instruction is coincident with a source register of a precedent instruction. The output-dependence occurs when the destination register is coincident between the subsequent instruction and the preceding instruction. These can be solved by holding the execution result of the subsequent instruction in a temporary register and writing data in an original register according to a program order. The execution of ~~a~~ an instruction using the execution result of the subsequent instruction can be started by reading out data from the temporary register even before the data are stored in the original register. This is called ~~as~~ "registry naming", and a temporary register used therefor is called ~~as~~ a

"remaining register".

In the out-of-order execution processor, a buffer for holding execution start (issue) waiting instructions such as a skipped instruction is called ~~as~~ a "reservation station", and a buffer for carrying out the rewriting of the execution result in program order is called ~~as~~ a "re-order buffer". As the size of the buffer is large, the range of ~~a~~ an instruction string for which the out-of-order execution can be performed can be ~~broadened~~ increased. Further, with respect to the control dependence, a branch destination is predicted and instruction execution is speculatively performed, whereby the parallel execution is enabled. The ~~cancel~~ cancellation of the speculative execution and ~~the~~ processing when an exception occurs are generally implemented as a function of the re-order buffer.

As described above, in order to ~~broad~~ increase the range of the instruction string for which the out-of-order execution can be performed, a re-order buffer having a large size is needed. A technological restriction is imposed on the upper limit of the size, and thus it is important how efficiently the re-order buffer having the restricted size is used. However, there has been hitherto such a disadvantage that even when an speculative instruction string in a re-order buffer is cancelled because of a failure of a branch prediction, an entry used for a non-completed load instruction which is contained in entries used for the speculative instruction string thus cancelled, cannot be immediately used for a subsequent instruction.

SUMMARY OF THE INVENTION

The present invention has an object to enhance the using efficiency of a re-order buffer.

In order to attain the above object, according to the present invention, there is provided a managing method for ~~an~~ a re-order buffer in an out-of-order execution processor for predicting the flow of a program by a branch prediction, ~~finding out~~determining a next executable instruction from ~~a~~ an instruction string in the program and speculatively executing the instruction on the basis of a dependence relationship between the prediction and the instruction, in which the re-order buffer rewrites an execution result according to a program order and the end of the instruction is notified from each of a plurality of function units containing a branching unit and a load unit to the re-order buffer by using a WRB number corresponding to an entry number of the re-order buffer, which comprises the steps of: managing the latest speculation state of a load instruction issued to the load unit by the load unit on the basis of a branch prediction success/failure signal output from the branching unit and suppressing notification to the re-order buffer by the load unit, as to a subsequent load instruction of a branch instruction for which the branching prediction has failed, on the basis of the WRB number of the subsequent load instruction even ~~when~~ if the processing of the load instruction concerned is finished, and re-using an entry stored with the subsequent instruction of the branching instruction which the branching-prediction has failed ~~branching instruction~~, by the re-order buffer, to store a new instruction before the end notification based on the WRB number of the entry concerned is received, wherein a control signal for discriminating non-speculative execution/speculative execution for every instruction, which corresponds to a branching level, is generated in an instruction fetch/decode unit, the branching level being set to zero when the

instruction concerned is an instruction for non-speculative execution, and the branching level being set at a value of 1 or more which is determined by the number of branching instructions interposed between the instruction for the non-speculative execution and the instruction for the speculative execution when the instruction concerned is an instruction for speculative execution, the control signal thus generated is held in said re-order buffer and said load unit, and the branching level is decremented by 1 in said re-order buffer and said load unit when a branch prediction failure signal is output from said branching unit, thereby managing the latest instruction speculation state.

Further, according to the present invention, there is provided an out-of-order execution processor for predicting the flow of a program by ~~a~~ branch prediction, ~~finding out~~determining a next executable instruction from ~~a~~ an instruction string in the program and speculatively executing the instruction on the basis of dependence relationship between the prediction and the instruction, in which a re-order buffer in the processor rewrites an execution result according to a program order and the end of the instruction is notified from each of a plurality of function units containing a branching unit and a load unit to the re-order buffer by using a WRB number corresponding to an entry number of the re-order buffer, which comprises: managing means for managing the latest speculation state of a load instruction issued to the load unit on the basis of a branch prediction success/failure signal output from the branching unit and suppressing notification to the re-order buffer, as to a subsequent load instruction of a branch instruction for which the branching prediction has failed, on the basis of the WRB number of the subsequent load instruction even ~~when~~ if the

processing of the load instruction concerned is finished, the managing means being contained in the load unit, wherein the re-order buffer re-uses an entry stored with the subsequent instruction of the branching instruction which the branching-prediction has failed ~~branching instruction~~, to store a new instruction before the end notification based on the WRB number of the entry concerned is received, wherein a control signal for discriminating non-speculative execution/speculative execution for every instruction, which corresponds to a branching level, is generated in an instruction fetch/decode unit, the branching level being set to zero when the instruction concerned is an instruction for non-speculative execution, and the branching level being set at a value of 1 or more which is determined by the number of branching instructions interposed between the instruction for the non-speculative execution and the instruction for the speculative execution when the instruction concerned is an instruction for speculative execution, the control signal thus generated is held in said re-order buffer and said managing means, and the branching level is decremented by 1 in said re-order buffer and said managing means when a branch-prediction failure signal is output from said branching unit, thereby managing the latest instruction speculation state.

According to the present invention, in the load unit, the latest speculation state of the load instruction issued to the self unit is managed on the basis of the branch-prediction success/failure signal output from the branching unit, and with respect to the subsequent load instruction of a branching instruction for which the branching prediction fails, the notification to the re-order buffer on the basis of the WRB number of the load

instruction concerned is suppressed even when the processing thereof is finished, so that the WRB number of the load instruction concerned can be prevented from being afterwards notified from the load unit even when a non-completed load instruction ~~exists~~ existing in the subsequent instruction string is cancelled. Therefore, even when a cancelled entry of the re-order buffer is re-used ~~for another~~ as a new instruction, the DONE bit of the another new instruction is prevented from being erroneously turned on, and the entry of the re-order buffer in which the subsequent instruction of the branching-prediction failed branching instruction is stored can be quickly re-used for storage of a new instruction without waiting for an end notification based on the WRB number of the entry concerned.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram showing the construction of a processor to which the present invention is applied;

Fig. 2 is a block diagram showing an embodiment of the processor to which the present invention is applied;

Fig. 3 is a diagram showing the construction of an entry of a re-order buffer;

Figs. 4A and 4B are diagrams showing a method of generating a branch level in a decoder;

Fig. 5 is a diagram showing the construction of an entry provided to a managing part of a load system; and

Fig. 6 is a diagram showing an example of the contents of a re-order buffer at ~~some~~ a given time.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Preferred embodiments according to the present invention will be described hereunder with reference to the accompanying drawings.

Fig. 1 is a block diagram showing the construction of a processor to which the present invention is applied.

The process shown in Fig. 1 is an out-of-order execution processor for predicting the flow of a program by branching prediction, searching a next executable instruction from ~~a~~ an instruction string in the program and speculatively executing ~~a~~ an instruction on the basis of a dependence relationship between the prediction and the instruction, ~~and it~~ which includes, as main parts, instruction fetch/decode unit 1, ~~some~~ function units, that is, branching unit 2, operation unit 3, store unit 4 and load unit 5, settling unit 7, and reservation station 8 provided at the front stage of the respective units. The settling unit 7 has re-order buffer 6 ~~having~~ which has plural entries 61. The load unit 5 has managing part 51 ~~having~~ which has plural entries 52 for holding and managing issued load instructions ~~issued~~.

In the embodiment of Fig. 1, the store unit 4 and the load unit 5 are designed as separate units, however, they may be ~~designed as~~ included in the same unit (load/store unit). Further, one reservation station 8 which is commonly used ~~to~~ along with the respective function units is provided. However, a reservation station may be provided ~~to~~ for each function unit, or a common reservation station may be provided to ~~each~~ a group of ~~some~~ function units.

The instruction fetch/decode unit 1 fetches ~~a~~ an instruction from ~~a~~ an instruction cache and decodes the instruction, ~~and~~. It also controls the

registry naming processing to overcome the instruction dependence relationship and the speculative execution in the branching direction predicted by branching prediction. In order to investigate the instruction dependence relationship, the instruction fetch/decode unit 1 refers to a preceding instruction which has ~~been~~ already been stored in the re-order buffer 6. Further, the instruction fetch/decode unit 1 ~~also~~ controls a pointer for indicating an empty entry 61 of the re-order buffer 6 to be ~~next~~ used next, and registers instruction information 63 such as the operation code of ~~a~~ an instruction and register information, ~~a~~ an instruction branching level 64, etc. into the empty entry 61. The branching level 64 is a control signal for discriminating non-speculative execution/speculation execution for every instruction. When the instruction concerned is ~~a~~ an instruction for ~~the~~ non-speculative execution, the branching level 64 is set to zero, and when the instruction concerned is ~~a~~ an instruction for ~~the~~ speculative-execution, the branching level 64 is set to a value which is equal to or more than 1 and determined by the number of branch instructions interposed between the instruction for the non-speculative execution and the instruction for the speculative execution. The entry 61 of the re-order buffer 6 is provided with V bit 62 indicating whether the entry concerned is in use or not, and the DONE bit 65 for indicating whether execution of the instruction of the entry concerned is finished or not. At the instruction registration time, the V bit 62 is ~~turned on~~ set to "1" to ~~thereby~~ indicate "in use", and the DONE bit 65 is ~~turned out~~ set to "0" to ~~thereby~~ indicate "no finish".

Further, the instruction fetch/decode unit 1 registers ~~into~~ with the reservation station 8 the decode result of each instruction together with the

branching level of the instruction and the WRB number of the entry 61 of the re-order buffer 6 in which the instruction is registered. When all the function units corresponding to the operand of the instruction registered are ~~allowed to be used~~ available, the reservation station 8 issues ~~the~~ an instruction to the corresponding function units of the branching unit 2, the operation unit 3, the store unit 4 and the load unit 5. At this time, the branching level and the WRB number are added to the instruction to be issued.

The branching unit 2 executes the issued branch instruction ~~thus issued~~, and outputs ~~the~~ a branch-prediction success/failure signal specifying the WRB number appendant to the branch instruction concerned to the instruction fetch/decode unit 1, the reservation station 8, the re-order buffer 7 and the load unit 5. In the instruction fetch/decode unit 1, a branch record is renewed on the basis of the branch-prediction success/failure signal. In the reservation station 8, when the branch prediction succeeds, the branch level of each instruction held is decremented by 1, and when the branch prediction fails, all the instructions for which the branch level is equal to or greater than 1 ~~or more~~ are cancelled. In the re-order buffer 6, when the branch prediction succeeds, the branch level 64 of each instruction held in the entry 61 is decremented by 1, and when the branch prediction fails, all the instructions for which the branch level is equal to or greater than 1 ~~or more~~ are cancelled and the V bit 62 of the entry 61 which is held is set to "0" to allow re-use. The processing of the load unit 5 receiving the branch prediction success/failure signal ~~will be described later~~ is described below.

The operation unit 3 executes an issued operation instruction, and at the time when the operation instruction is finished, it specifies the WRB

number appendant to the operation instruction and transmits an end notification to the re-order buffer 6. The re-order buffer 6 turns on the DONE bit 65 of the entry 61 of the WRB number thus specified to "1" (indicating the end of the instruction).

The store unit 4 transmits a store request to the main memory according to an issued store instruction, specifies the WRB number appendant to the store instruction concerned when the store instruction is finished, and transmits an end notification to the re-order buffer 6. The re-order buffer 6 turns on the DONE bit 65 of ~~the~~ entry 61 of the WRB number thus ~~specified~~ setting it to "1" (indicating the end of the instruction).

The load unit 5 is equipped with a managing part 51 having plural entries 52, each of which holds a load instruction 53 issued to the ~~self~~ lead unit 5 together with the branch level 54 and WRB number 55 appendant thereto. Upon receiving the branch prediction success/failure signal from the branch unit 2, the managing part 51 decrements the branch level 54 of the load instruction held in the entry 52 by 1 when the branch prediction succeeds, thereby managing the latest speculation state of each load instruction. When the branch prediction fails, the managing part 51 sets the entries 52 having the branch level 54 of 1 or more to ~~the cancel~~ a canceled state. The load instructions registered in the respective entries 52 of the managing part 51 are processed in ~~issuance~~ the order ~~thereof~~ of issuance, and the processing for transmission of cache data from a data cache (not shown), transmission of a load request to the main memory in the case of a mis-hit, writing of the replay data to the load request into the data cache are all executed. If the entry 52 for storing a load instruction is not set to the ~~cancel~~

canceled state at the execution end time of the load instruction concerned, the WRB number 55 stored in the entry 52 is specified and ~~the~~ an end notification is transmitted to the re-order buffer 6. The re-order buffer 6 ~~turns on~~ sets the DONE bit 65 of the entry 61 of the specified WRB number to "1" (indicating the instruction end). On the other hand, if the entry 52 for storing a load instruction for which the execution is finished is set to the ~~cancel~~ canceled state, the end notification to the re-order buffer 6 on the basis of the WRB number 55 is suppressed.

The settling unit 7 records all the non-completed instructions ~~into~~ to the entries 61 of the re-order buffer 6, and receives the end notification specifying the WRB number from the respective function units 2 to 5 to ~~turn on~~ set the DONE bit 65 of the corresponding entry 61 to "1", thereby managing the presence or absence of the execution end of each instruction. ~~Many~~A plurality of entries 61 for holding non-completed instructions are provided in re-order buffer 6, and a physical number is given to each entry 61 to ~~thereby~~ enable each entry to be uniquely identified. The number given to each entry 61 is called ~~as~~ a "WRB number". ~~In each~~Each entry 61 ~~is registered~~ contains instruction information 63 ~~containing~~ which includes the operation code of ~~a~~ an instruction decoded by ~~a~~ an instruction fetch/decode unit 1, register information and ~~registered also~~ DONE bit 65 indicating whether execution of ~~a~~ an instruction registered in each entry 61 is ~~finished or not~~ has been completed. This bit is called ~~as~~ a "DONE bit". The DONE bit 65 is turned on at the time when the instruction concerned is finished. With respect to ~~a~~ an instruction for which the DONE bit 65 is turned on because the instruction operation in the corresponding function unit is completed, the

branch related with the instruction is settled and all the previously-issued instructions are settled, the settling unit 7 carries out the processing of writing the execution result of the instruction concerned into a software-visible register or the like. When the branch prediction fails, all the results of the subsequent instructions of the branch concerned are invalidated, and all the instructions under the speculation state at the present time are cancelled from the re-order buffer 6. In order to manage which instruction the speculation instruction at the present time corresponds to, the settling unit 7 decrements the branch level 64 of each instruction held in the entry 61 by 1 when the branch prediction success is notified from the branch unit 2, thereby managing the latest speculation state of each instruction. Accordingly, when receiving the branch prediction failure signal from the branch unit 2, instructions having the branch level of 1 or more are speculation instructions at the present time, and all the speculation instructions are cancelled. In addition, the V bits 62 of the entries 61 holding these speculation instructions are set to "0", thereby enabling their re-use ~~thereof~~.

~~[Embodiments]~~Description of Embodiments

Fig. 2 is a block diagram showing an embodiment of a processor ~~to which~~ of the present invention ~~is applied~~. The processor of this embodiment includes instruction cache 100, decoder 101, load/store controller 102, branch controller 103, re-order buffer (ROB) 104, reservation stations (RS) 105-1 and 105-2, cache controller 106, data cache 107, bypass controller 108, operator 109, request/reply controller 110, renaming register (RR) 111, scalar register (SR) 112, instruction address counter 113, branch prediction part 114 and managing part 115 as main parts, and ~~perform~~ performs parallel execution of

instructions through ~~a~~ an instruction pipeline, particularly out-of-order execution.

~~In the relationship of~~ With respect to Fig. 1, the instruction address counter 113, the instruction cache 100, the decoder 101 and the branch prediction part 114 correspond to the instruction fetch/decode unit 1 ~~of Fig. 1~~, the reservation stations 105-1 and 105-2 correspond to the reservation station 8 ~~of Fig. 1~~, the load/store controller 102, the cache controller 106, the data cache 107, the request/reply controller 110 and the managing part 115 correspond to the store unit 4 and the load unit 5 ~~of Fig. 1~~, the branch controller 103 corresponds to the branch unit 2 ~~of Fig. 1~~, the re-order buffer 104 corresponds to the settling unit 7 having the re-order buffer 6 ~~of Fig. 1~~, and the bypass controller 108 and the operator 109 correspond to the operation unit 3 ~~of Fig. 1~~. The renaming register 111 and the store register 112 which are not shown in Fig. 1 are illustrated in Fig. 2.

The renaming register 111 contains a register which is in one-to-one correspondence ~~to~~ with the entry of the re-order buffer 104, and the result of an operating instruction, a load instruction or the like stored in an entry having WRB number ~~i~~ "i" of the re-order buffer 104 is stored in the register corresponding to the WRB number i in the renaming register 111. Thereafter, when the instruction stored in the entry of the WRB number i of the re-order buffer 104 is settled, the scalar register number stored in the entry concerned is output to a path 133, and the register corresponding to the WRB number i in the renaming register 111 is written into the register specified by the scalar register number in the scalar register 112 through a path 134.

The construction and operation of this embodiment will be described

in detail below.

The instruction address specified by the instruction address counter 113 is given through path 137 to the instruction cache 100 and the branch prediction part 114. The branch prediction part 114 receives ~~a~~ an instruction address specified by the instruction address counter 113 to judge whether the branch record of a branch instruction indicated by the instruction address concerned is or is not registered ~~or not~~. If it is registered, success/failure of branching is predicted on the basis of a past registered branch record ~~registered~~. If the branch success is predicted, a branch destination address is transmitted to the instruction address counter 113 through path 138 to renew the instruction address counter 113. In the case where the branch fails and in the case where the instruction is out of the branch instruction, the counter value of the instruction address counter 112 is incremented by every amount of one instruction.

The decoder 101 decodes ~~a~~ an instruction fetched from the instruction cache 100 through path 120, carries out the processing to suppress hazards caused by a register number while devising allocation of the renaming register 111 in consideration of preceding instructions accumulated in the re-order buffer 104, and registers information ~~of~~ from decoded instructions or the like into the re-order buffer 104 in the instruction order of the program. In the case of an operation instruction, the decoder 101 registers the decoded data into the reservation station 105-1 of the operation system. In the case of a branch instruction or a load/store instruction, the decoder 101 registers the decoded data into the reservation station 105-2.

Fig. 3 shows the construction of the entry of the re-order buffer 104.

WRB number 1042 for uniquely identifying the entry thereof is physically given to each entry 1041. ~~In each~~Each entry 1041 ~~are stored~~ contains V bit 1043 for indicating whether the entry concerned is in use or non-use, instruction operation code 1044, scalar register number (SR number) 1045 for storing the result of the instruction concerned, control information 1046, branch level 1047 and DONE bit 1048 for indicating whether the instruction concerned is or is not finished ~~or not~~. The contents of the entries of the reservation stations 105-1 and 105-2 are substantially the same as the entries of the re-order buffer 104.

The branch level 1047 is generated in such a manner as shown in Fig. 4A in the decoder 101. The decoder 101 has branch level counter 1011, and the initial value ~~thereof~~ is set to zero. ~~Under~~In this state, instructions are successively decoded, and the current count value 0 of the branch level counter 1011 is allocated to each instruction as branch level 0. When one branch instruction is decoded, the branch level counter 1011 is incremented by 1, and the branch level "1" is allocated to the subsequent instructions of the branch instruction concerned. Further, when another ~~one~~ branch instruction is decoded, the branch level counter 1011 is incremented by 1 ~~again~~. When the decoded branch instruction ~~thus decoded~~ is executed in the branch controller 103 and the branch prediction success/failure signal associated with the branch instruction concerned is output to the path 123, the decoder 101 decrements the branch level counter 1011 by 1 ~~when~~ if the branch prediction succeeds. On the other hand, ~~when~~ if the branch prediction fails, the decoder 101 clears the branch level counter 1011 and sets it to 0.

When plural instructions from load instruction 1 ~~until~~ through

branch instruction 3 as shown in Fig. 4B are fetched from the instruction cache 100, the decoder 101 allocates the branching levels as shown in Fig. 4B to the respective instructions, and registers the instructions into the re-order buffer 104 through path 121. The operation instructions ~~out of~~ from these instructions are also registered in the reservation station 105-1, and the load instructions, the store instructions and the branch instructions are also registered in the reservation station 105-2. In the re-order buffer 104 and the reservation stations 105-1 and 105-2, when the branch prediction success/failure signal are output from the branch controller to the path 123, all the branch levels of entries having ~~the~~ a branch level of 1 or more are decremented by 1 ~~when~~ if the branch prediction succeeds, and all the instructions of entries having the branch level of 1 or more are cancelled ~~when~~ if the branch prediction fails.

The control information 1046 ~~of~~ in Fig. 3 is mainly information related with operands. In the case of ~~a~~ an instruction having a dependence relationship with a preceding instruction, the WRB number of an entry of the re-order buffer 104 in which the preceding instruction is stored is ~~put~~ included in the control information 1046. For example, in the case of a subsequent instruction having as a source the execution result of ~~a~~ an instruction stored in an entry having WRB number ~~=~~ equal to 10 of the re-order buffer 104, the WRB number ~~=~~ of 10 is ~~put~~ included in the control information 1046. When all the execution results of the instructions stored in the entries having WRB numbers described in the control information 1046 are allowed to be used, preparation of the operands of the instructions of the entries concerned is completed. Therefore, if there are empty resources of the

corresponding function units, the instructions are issued from the reservation stations 105-1 and 105-2.

Next, the construction and operation of this embodiment will be described while dividing into a load/store system, an operation system and a branching system.

~~[Load/store system]~~Description of the Load/Store System

Upon receiving a load instruction and a store instruction from the reservation station 105-2 through path 140, the load/store controller 102 transmits the instructions through path 122 to the cache controller 106 and the data cache 107, and transmits the data to the managing part 115.

The managing part 115 has plural entries for holding issued load instructions and store instructions until the processing of the instructions is completed, and registers into empty entries the load instructions and store instructions transmitted from the load/store controller 102 through the path 122 to manage these instructions.

Fig. 5 shows the construction of entry 1151 provided to the managing part 115. Tag 1152 is an identifier attached when the load instruction or store instruction registered in the entry concerned is transmitted to the main memory, and it is preset. Since the same tag is returned when a reply is returned from the main memory, the tag 1152 is used to discriminate which instruction corresponds to the reply ~~corresponds to~~. V bit 1153 indicates whether the entry concerned is in use or ~~non-use~~ not used, and it is ~~turned on~~ set to "1" when it is used, and it is ~~turned off~~ set to "0" when it is not used. An operation code 1154, branch level 1155 and WRB number 1156 are the operation code of the load instruction or store instruction thus issued, and the

branch level and WRB number ~~appendant~~ are appended to the instruction, respectively. Request bit 1157 is a bit for managing whether the request of the load instruction or store instruction concerned is or is not output to the main memory ~~or not~~, and it is ~~turned on~~ set to "1" at the time when the request is transmitted to the main memory. Cancel bit 1158 is ~~turned on~~ set to "1" ~~when~~ if the load instruction or store instruction concerned is cancelled.

In the case of ~~the~~ a load instruction, the data cache 107 ~~makes an judgement on hit/mistake~~ judges a hit or miss, and in the case of "hit", the data cache 107 transmits the cache data through the path 127 to the bypass controller 108, and notifies ~~the~~ this fact through a path 142 to the managing part 115 while specifying the WRB number ~~appendant~~ appended to the load instruction concerned. If the cancel bit 1158 of the entry 1151 having the same WRB number is set to "0", the managing part 115 transmits an end notification specifying the WRB number through path 128 to the re-order buffer 104 and the reservation station 105-1 and 105-2. If the cancel bit 1158 is set to "1", ~~no such~~ an end notification is not transmitted.

The data cache 107 transmits a ~~mistake signal~~ miss-signal through path 139 to the cache controller 106 when ~~it makes~~ a cache-~~mistake~~miss is made. The cache controller 106 transmits a cache-~~mistake~~miss request through path 125 to the request/reply controller 110 while specifying the WRB number. If the cancel bit 1158 of the entry 1151 having the specified WRB number in the managing part 115 is not set to "1", the request/reply controller 110 transmits a load request through path 135 to the main memory (not shown) with ~~appending~~ appended tag 1152. When reply data having the tag appended thereto is returned from the main memory through the path

136, the ~~request /reply~~ request/reply controller 110 refers to the cancel bit 1158 of the entry 1151 having the same tag appended thereto in the managing part 115 through path 144. If it is not set to "1", the data ~~are~~ is written into the data cache 107 through the path 126, and also transmitted out to the bypass controller 108 and the renaming register 111 through the path 127. If the cancel bit 1158 is set to "0", ~~no such an~~ this operation is not carried out. In any case, the tag is ~~specified~~ provided to the managing part 115, and the end of the load instruction is notified. If the cancel bit 1158 of the entry 1151 having the same tag appended thereto is set to "0", the managing part 115 transmits the end notification specifying the WRB number 1156 of the entry concerned through path 128 to the re-order buffer 104 and reservation stations 105-1 and 105-2. However, if the cancel bit 1158 is set to "1", ~~no such~~ an end notification is not transmitted.

In the case of the store instruction, the cache controller 106 transmits a store request through the path 125 to the request/reply controller 110 ~~with~~ while specifying the WRB number irrespective of ~~hit/mistake~~ a hit or miss of the data cache 107. The request/rely controller 110 refers to the entry 1151 having the specified WRB number from the managing part 115 through the path 144, and makes a store request through the path 135 to the main memory with the tag 1152 ~~thereof being~~ attached thereto. When a reply is returned from the main memory, the completion of the store instruction is notified to the managing part 115 while specifying the tag. If the cancel bit 1158 of the entry 1151 having the same tag appended thereto is set to "0", the managing part 115 transmits the end notification specifying the WRB number 1156 of the entry concerned through the path 128 to the re-order

buffer 104 and the reservation stations 105-1 and 105-2. However, if the cancel bit 1158 is set to "1", ~~no such~~ an end notification is not transmitted.

~~[Operating system]~~Description of the Operating System

When an operating instruction is issued from the reservation station 105-1 through the path 124, the bypass controller 108 transmits an operating instruction issued from the reservation station 105-1 through path 131 to the operator 109 to perform an operation by using data transmitted from the renaming register 111 through path 129, data transmitted from the scalar register 112 through path 130 and data transmitted from the data cache 107 through path 127. The operation result achieved by the operator 109 is registered in accordance with the WRB number appended to the operation instruction concerned in the renaming register 111 through path 132.

The reservation station 105-1 of the operating system ~~itself~~ manages ~~how many~~ the number of cycles ~~are~~ needed to finish the instruction after issuance of the operation instruction, and it transmits ~~the~~ an end notification to the re-order buffer 104 and the reservation station 105-2 through path 145 while specifying the WRB number every time the instruction is finished.

~~[Branching system]~~Description of the Branching System

Upon receiving a branch instruction from the reservation station 105-2 through path 141, the branching controller 103 makes a branch judgment of the branch instruction concerned, and transmits a branch prediction success/failure signal having a WRB number appended thereto through the path 123 to the branch prediction part 114, the decoder 101, the re-order buffer 104, the reservation stations 105-1, 105-2 and the managing part 115. The branch prediction part 114 renews the branch record on the

basis of the branch prediction success/failure signal. In each of the re-order buffer 104 and the reservation stations 105-1 and 105-2, the branch level of each instruction held in the entry thereof is decremented by 1 ~~when~~, if the branch prediction succeeds, and all the instructions having the branch level of 1 or more are cancelled ~~when~~ if the branch prediction fails. ~~When~~If the branch prediction succeeds, the managing part 115 decrements the branch level 1155 of the instruction held in the entry 1151 thereof by 1, and ~~when~~ if the branch prediction fails, the managing part 115 sets the cancel bits 1158 of the entries 1151 holding the instructions having the branch level of 1 or more to "0", thereby setting the entries concerned to the ~~cancel~~ canceled state.

Fig. 6 is a diagram showing the re-order buffer 104 at the time when the execution is finished until the store instruction 1 of the instruction string shown in Fig. 4B. Thereafter, the branch instruction 1 is executed, and the operation instruction 2, the operation instruction 3, the load instruction 3, the load instruction 4, the operation instruction 4, etc. of the subsequent instructions are speculatively executed while ~~the~~ a branch judgment is made in the branch controller 103. If the branch prediction failure of the branch instruction 1 is settled, all the instructions having the branch level of 1 or more are cancelled, and all the entries used therefor are re-used by the subsequent instructions in the branch direction determined by the branch instruction 1. In the prior art, the entries of the load instruction 3, the load instruction 4, etc. are not ~~usable~~ available until the ~~end~~ notification ~~indicating~~ of the WRB numbers of the entries ~~comes~~ is completed.

As described above, according to the present invention, the entries of the re-order buffer, in which the subsequent instructions of ~~a~~ the branch

instruction which the branch-prediction has failed ~~branch instruction~~ are stored, can be re-used for storage of new instructions without waiting for the end notification based on the WRB numbers of the entries concerned. Therefore, the entries of the re-order buffer can be effectively used, and thus the range of an out-of-order executable instruction string can be enlarged.